Advances in Computer Science Research (ACRS), volume 54
International Conference on Computer Networks and Communication Technology (CNCT2016)

Mining Cross Site Scripting Vulnerabilities Based on HTML5 in Email Systems

Jian-zhong ZHANG[1] and Ao CHAI[2]

[1]College of Computer and Control Engineering, Nankai University, China

[2]College of Computer and Control Engineering, Nankai University, China

[1]zhangjz@nankai.edu.cn, [2]chaiao12@163.com

Keywords: HTML5, XSS, Web Security

Abstract. Cross-site scripting attacks has always been one of the most common attacks to the front-end network applications. With the popularity of HTML5, the security of Email systems is facing new challenges. In this paper, we propose a new approach which utilizes HTML5 new tags and new attributes to construct storage-type XSS attack vectors. Based on this method, we have tested several domestic and foreign common mailbox and detected six HTML5-based XSS vulnerabilities. The final evaluation results show that our method can detect storage-type XSS vulnerabilities based on HTML5 effectively.

Introduction

XSS (Cross-site scripting) allows attackers to inject client-side scripts. It can cause serious injury to users' privacy. According to Symantec Security Vulnerabilities Document [1], XSS vulnerabilities account for about 84% of all web vulnerabilities. Still, in the newest Symantec Internet Security Threat Report [2], it listed XSS as No.5 of "Top 10 Vulnerabilities Found Unpatched on Scanned Web Servers". Meanwhile, it is also evaluated by OWASP (Open Web Application Security Project) as one of the top ten security vulnerabilities (ranked third) [3].

Depending on the XSS (Cross-site scripting) characteristics and methods of attack, generally, the security workers divide them into three main types:

- Reflected XSS Attack
- Stored XSS Attack
- DOM-based XSS Attack

Among those types, Stored XSS Attack can still cause serious damages to Webmail systems.

Security researchers have been repeatedly reporting that the mail system eYou has XSS vulnerabilities in the main body of the emails. The famous Gmail was also exposed recently that it contained XSS vulnerabilities.[4] At the beginning of 2016, a security expert from California, U.S.A., has discovered and disclosed a high-risk vulnerability in the old brand Yahoo! mail system.[5] The vulnerability can forward all of user's messages to the attacker's mailbox without victim's awareness. Apparently, there's never a stop with the XSS attack.

In addition, in the HTML4 Era, many workers went through a lot of experiments in order to detect the presence of Web System XSS security vulnerabilities. They did the research of XSS attack vectors, and summarized them into an attack scripts list (XSS cheat sheet). Among those lists, there are several famous ones, such as ha.ckers.org[6],

which was released by the renowned security engineer Rsnake, xssed.com[7], which is organized by Kevin Fernandez, and Prevention Cheat Sheet 2016 [8], which is maintained by OWASP.

Due to the great destructiveness of XSS, many security researchers began to study various prevention methods [9-11]. As in 2010, Bateset al proposed a new filtering technology. This technology can effectively prevent the operation of harmful script in HTML. Thereafter, this technology has been applied to the Google Chrome Browser [12]. Meanwhile, Microsoft, in its browser IE8 and later versions, limited the operations of the local JavaScript files, and remove a lot of support for the pseudo-protocol [13]. Obviously, blacklist-based filters are generally used by the majority of browsers.

However, the new version of the browsers began to be fully compatible with HTML5. With the prevalence of HTML5, the attackers can still construct more attack vectors. HTML5 involves plenty of new tags, attributes and events. These can help the attackers bypass the filtering mechanism based on the black list policy [14]. New tags such as <video><audio> and so on, new attributes such as autofocus, which can be used to trigger events such as onblur, onfocus [15]. Furthermore, HTML5 is platform-independent, which means it can be implemented in many browsers. As a result, it is more likely that some attack vectors would not be triggered in some browsers, but are triggered in other browsers. This increased the difficulty of testing XSS significantly. In fact, there are many E-mail system filters does not contain the ability to detect the attack vectors based on HTML5, or only has the limited ability. This leads to the existence of blind spots in HTML5 security testing [16].

Now, there are a number of attack script lists for HTML5 based XSS. One of the most famous ones is the HTML5 Security Cheat Sheet [17]. The list includes plenty of the XSS attack vectors. However, some of them are relatively basic vectors, which means the list doesn't consider the combinations and variations.

Considering the present security situation of the mail system as well as the new problems and challenges brought by HTML5, we propose a systematic approach to detect the XSS vulnerability. In part II, we firstly analyzed the characteristics of the new tags and attributes which may trigger XSS vulnerabilities in HTML5. Afterwards, we introduced a systematic way to construct an attack vector set, which was then added to the test message. According to the different check points, we used a detection tool to send the test messages to several domestic and foreign mail systems, by detecting their filtering mechanisms to see if they were able to protect the integrity of the attack surface and filter all the attack vectors. In part III, by applying our tool, we have found several storage XSS vulnerabilities in some domestic and foreign mail systems in the experiment, which proved that our approach is capable of detecting the potential XSS vulnerabilities in mail systems. Finally, the conclusion is drawn in part IV.

Approach

Our detection can be mainly divided into three stages. The first stage is to summarize all HTML5 that can trigger the XSS tags, attributes and events, and then, make use of these tags and attributes to construct the basic attack vectors; the second stage is to vary the attack vectors based on this basic attack vector set; at last, insert the varied attack vectors into a test email, and send the email to the targeted mail systems, then check to see if the vectors are filtered at each check point.

A. Establish HTML5 Attack Vector Set

HTML5 is the fifth major revolution in the hypertext markup language, which is officially completed by the end of October 2014. It contains a lot of practical functions, greatly enhances the network multimedia features. To this end, it also adds 27 new tags and 30 new attributes [18]. However, HTML5's new edition does not mean it is safe [18], on the contrary, it will give attackers a lot of new ways of attack. For example, in these new tags, there are more than 10 labels and at least 6 attributes that are very vulnerable to XSS attacks (see Table 1). These tags have their own "attributes", and these attributes can be used by attackers to insert the related XSS attack vectors.

Table 1. New tags and attributes that can trigger XSS attacks

	Attack vectors
HTML5 Tags	<article><aside><audio><embed><figcaption><figure><footer> <header><mark><meter><nav><output><progress><section> <source><summary><video>
HTML5 Attributes	autofocus, formaction, onhashchange, onformchange, onforminput, srcdoc

In addition, the Scalable Vector Graphics tag <svg>, which is newly supported by HTML5, can also be injected with XSS attck vector. The tag <svg> can be inserted into some HTML5 new tags, such as <embed>, <object> and <iframe>.

In summary, we constructed three kinds of primitive vectors. These vectors include new tags, new events and the use of scalable vector graphics tag (see Table 2). These original basic vectors we just created will trigger XSS, but the form of which is so simple that they hardly bypass the filters. Afterward, we have them varied, and try to bypass the filter.

Table 2. The primitive attack vectors

Categories	Primitive vectors
With new tags	<video><source onerror=alert(1)>
	<audio><source onerror=alert(1)>
	<article onclick=alert(1)>ARTICLE</article>
With new events	<input autofocus onfocus=alert(1)>
	<body onhashchange=alert(1)><a href=#>CLICK
With <SVG>	<svg onload=alert(1)>

B. Transform the primitive attack vector sets

Email server systems nowadays generally use the keyword blacklist strategy and the regular expression to filter the input data [19]. This way, although it's effective, can still not resist the various transformations of all kinds of tags.

There are three main approaches to the variation of attack vectors:

1. Original Vectors Encoding

With different encoding mechanism, original vectors can be combined into variety forms malicious attack vectors. so the encoded vectors can bypass the filter and inject into the mail server with certain probability.

Here we should consider the three main parts:

- HTML entity encoding
- Tag attribute encoding
- Special attribute data

The entity coding can be divided into two types: "&entity_name" and "&#entity_number;". The former is relatively fixed, while the latter can be achieved in two ways: decimal and hexadecimal. For example, the encoding results of tag "<script>" are: "<script>", "<script>(dec)" and "<script>(hex)".

We can still encode the attributes using backslash followed by encoding. This time, it can be encoded into octal, decimal, hexadecimal and Unicode. We also take the label "<script>" for instance, the results are: "<scri\160t> (OCT)", "<scri\112t> (DEC)", "<scri\x70t> (HEX)" and "<scri\u0070t> (Unicode)".

Finally, the BASE64 encoding mechanism can be used to encode the special attribute data.

2. Confuse the special characters within the original vectors

- Change the letter case in tags: the letter case in the tags does not affect the operation of the code, but it may confuse the filter's identification of keywords.
- Insert newlines or tabs: as above, the browsers will skip these special characters in the interpretation and operation of the code, meanwhile, the special characters may also bypass the filters.
- Change or add quotation marks: with respect to the attribute values within a tag, there are usually four ways to indicate: with single quotes, with double quotes, with anti-quotes, and without any quotes. To use the mixed or unpaired quotes can easily lead to the result that some events successfully bypass the filter and become a new attribute itself.
- Change to slash space: in the most cases, a slash can play the same function as a space in HTML, but it can easily bypass filter.

3. Recombine the original vectors:

The attack vectors can be nested within other vectors by the attackers, then try to bypass the filter. Many filters ignore the content in an attribute. Even if the vectors are being checked, the filters focus more on whether it's pseudo protocol in the attribute (such as JavaScript pseudo protocol). As for some tags, because of their own characteristics, it's very often to insert other labels into their interior and even attributes. Such as <embed>, <iframe> and <object>. This will cause a very large disturbance to the filters who use the black list strategy. The approach is usually to add and delete the angle brackets from tags,

resulting in a content overflow and then bypassing the filter. Such as the new attribute "srcdoc" in HTML5, in which it specifies the content displayed in the page framework, but you can insert a new label, triggering a XSS attack.

According to the statement above, we construct an attack vector set, as shown in Table 3.

Table 3

Categories	Attack vectors	Variation Approach
With new tags	<video onerror=`alert(1)`><source></source></video>	Confusing
	<audio/onerror=alert(1)><source></source></audio>	
	<video style="display:none;" onerror="s=document.createElement('script'); s.src=`javascript:alert(1)`; document.body.appendChild(s);"src="/" height="0" width="0"></video>	
	<IMG SRC=&#0000106&#0000097&#0000118&#0000097&#0000115&#0000099&#0000114&#0000105&#0000112&#0000116&#0000058&#0000097&#0000108&#0000101&#0000114&#0000116&#0000040&#0000039&#0000088&#0000083&#0000083&#0000039&#0000041>	Encoding
With new events	<a OnMoUsEoVer="alert(/XSS/)">MOUSE	Confusing
	<InPuT oNfOcus=alert(1) autofocus>	
	<input onblur=alert(1) autofocus><input autofocus>	
	<form><button formaction="javascript:alert(1)">BUTTON</button></form>	Recombine
	<body oninput=alert(1)><input autofocus>	
With <SVG>	<iframe srcdoc="<img src=x:x onerror=alert(1)>" />	Encoding
	<svg//onLoad="s=document.createElement('script');s.src='http://121.42.167.152/tom/tom.js';document.body.appendChild(s);"></svg>	
	<iframe srcdoc="<svg onload=alert(/XSS/)>"></iframe>	Recombine
	<svg xmlns="http://www.w3.org/"><axmlns:xlink= "http://www.w3.org/" xlink:href="javascript:alert(1)"><rect width="100" height="100" fill="white"/></svg>	

C. Selection of the attack points

The attack points of a mail system mainly consist of five parts, the attacker can inject XSS into each part of the mail [20-21]. Therefore, in order to perform a comprehensive examination of the filtering mechanism of the mail system, according to the composition of the mail [22], we need to perform an adequate test to all the five parts which offer email users of access of input. These five parts are:

- Email Subject:

Email subject can appear directly in the mailing list, so the cost of attack through the subject is the lowest. Generally speaking, most mail servers do not support the rich

text format for the message title. However, there are still a number of commonly used mail system is detected XSS vulnerability in the subject.

- Email Body:
  This part is the focus of attackers, because nearly all of the email servers support rich text format in this area. Therefore, the email body suffers the hardest hit from attackers.
- Attachment Title:
  Due to the fact that Windows does not support the use of angle brackets in a file's name, so it is often easy to ignore the check of attachment titles. However, Linux systems allow a file name with a pair of angle brackets. Therefore, we must take it into consideration.
- Attachment Body:
  Most mail servers allow users to preview the content of the attachments. If the attachments involve malicious script code, they will cause attacks on users.
- Sender Name:
  Like the attachment title, the name of the sender is also easily overlooked by the security workers.

Experiment

Based on the work above, we apply a dynamic XSS vulnerability detection tool. It can write the attack vector set in different parts of the message, namely the different check points, then connect to a SMTP server. After verification, the detection tool sends the testing message, which contains the attack vector set, to the mail server.

The detection tool is written in Python Language, using the smtplib module [23]. The module can quickly establish the mail object, SMTP session connection and use obj.sendmail() to send a testing message to the mail server to be detected.

Experimental flow chart is shown in Figure 1.

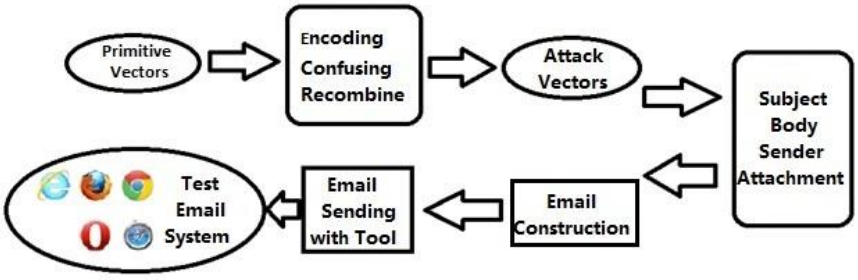

Figure 1. Experimental flow

Because most of the mail systems have the function of judging whether an Email is a spam or not, so we need to consider two strategies, and apply them into this detection tool, so that our testing messages will not be detected as spams.

- Set a threshold of mail-sending frequency
  Do not use the same email address sending messages to the same email system frequently. Some Email systems consider of the reasons of security, they don't accept emails from the same mail address in a short period of time. These messages will be automatically attributed to spams. There are two solutions: first, to set a transmission interval to the detection tool, like sending the testing messages to the same mailbox

every 15 seconds; second, to increase efficiency, sending the testing messages to the targeted mailbox with different email addresses.

- Attack vector confusion
  Don't compose the messages only with the attack vectors. If some email systems detect there are only a large number of HTML tags and no text in a single message, they will define it as a spam directly due to the security considerations. Therefore, the text format content and the HTML tags vectors must be in accordance with a certain proportion. The proportion will be a tiny different depending on the specific email systems, so the testing tool needs to have a function of constructing the message main body automatically.

Due to the different HTML parsing approaches applied by different browsers, there will be the case that the same attack vector can be triggered in different browsers. So we should experiment our testing tools with various mainstream browsers in the seventh step.

We implement our tool in some domestic and foreign commonly used mail systems. A total of 6 exploitable stored XSS vulnerabilities has been found. For example, with our tools, we found a vulnerability of <iframe> tag below:

```
<iframe srcdoc="<svg onload=alert(/XSS/)>"></iframe>
```

This vulnerability exists in a very widely used corporate mail server, and there are a lot of clients using it. The filtering mechanism of this email system is not strict to detect the new HTML5 tag <svg> or the content within the new attribute "srcdoc". When we write the malicious code in the event part of the <svg> tag and write the tag into the srcdoc attribute of the <iframe>, the attack vector will bypass the filter. As a result, trigger the XSS attack.

Another example was that we injected the code below into an attachment, and sent it to a foreign old brand mail server provider. If the user previews the attachment, XSS will be triggered.

```
<pre>
<html>
<head></head>
<body>
<something:script
xmlns:something="http://www.w3.org/">alert(/XSS/)</something:script>
</body>
<script>alert(/XSS/)</script>
</html>
</pre>
```

While most of the mail servers have their own filtering mechanisms, these mechanisms have not fully filtered the new labels and new attributes yet.

Conclusion

Until now, XSS attacks still pose a threat to the sensitive user information. There are still a lot of problems with the black list strategy to filter the attack vectors. The two main

reasons are, the security staff update the blacklist not timely, or it is difficult to think comprehensively about all kinds of variation in a same tag.

We perform a systematically test to some mail systems on stored XSS. The key to our approach is to construct a new set of XSS attack vectors by using the new tags and attributes of HTML5. After that, we apply our tools to some of the commonly used domestic and foreign e-mail system, and a total of 6 exploitable stored XSS vulnerabilities are found. The evaluation result shows the effectiveness of our method and tool.

References

[1] Symantec Internet Security Threat Report: Trends for July December 2007 (Executive Summary)

[2] Internet Security Threat Report Internet Report VOLUME 21, APRIL 2016

[3] Top 10 2013 [Online]: https://www.owasp.org/index.php/Top_10_2013

[4] http://blog.bentkowski.info/2014/06/gmail-and-google-tale-of-two-xss-es.html

[5] http://www.infosecurity-magazine.com/news/yahoo-mail-patches-severe-xss-flaw/

[6] http://ha.ckers.org/xss.html

[7] http://xssed.com/contact

[8] OWASP XSS Cheat Sheet 2016: https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

[9] Weinberger, P. Saxena, D. Akhawe, M. Finifter, R. Shin, and D. Song. "A systematic analysis of XSS sanitization in Web application frameworks" In ESORICS, 2011.

[10] D.Kavitha, M.R.Akshaya, M.Karthick, K.Baghya, K.Gomathi Raja Eswari "Prevention of CSRF and XSS Security Attacks in Web Based Applications" IJIRSET Vol. 5, Issue 3, March 2016

[11] Monika Rohilla, Rakesh Kumar, Girdhar Gopal "XSS Attacks: Analysis, Prevention & Detection" IJARCSSE Volume 6, Issue 6, June 2016

[12] Daniel Bates, Adam Barth, Collin Jackson, "Regular expressions considered harmful in client-side XSS filters", Proceedings of the 19th international conference on World wide web, April 26-30, 2010, Raleigh, North Carolina, USA

[13] http://blogs.msdn.com/b/ie/archive/2008/07/02/ie8-security-part-ivthe-xss-filter.aspx

[14] Rafay Baloch, "HTML5 Modern Day Attack and Defence Vectors", 2014 RHA InfoSEC

[15] Lavakumar Kuppan, "Attacking with HTML5"

[16] Bosong SUN, Ali Abbasi, Jianwei ZHUGE, Haixin DUAN, Heng WANG “Computer Applicaitons and Software” 2013 (In Chinese)

[17] https://html5sec.org/

[18] W3C[Official Website]: https://www.w3.org/TR/html/single-page.html

[19]https://www.owasp.org/index.php/Testing_for_Stored_Cross_site_scripting_%28OTG-INPVAL-002%29

[20] Persistent XSS in Rediffmail.com Email Subject Line: http:// nishant.daspatnaik.com / rediff_ subject.php

[21] WooYun-2015-161275: http://www.wooyun.org/bugs/wooyun-2015-0161275

[22] RFC 822[Online]: http://www.ietf.org/rfc/rfc0822.txt

[23] Al Sweigart, "Automate the boring stuff with python", 2015